

AD-A194 454

A PROOF OF THE KAHN PRINCIPLE FOR INPUT/OUTPUT AUTOMATA 1/1

(U) MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR
COMPUTER SCIENCE M A LYNCH ET AL. JAN 88

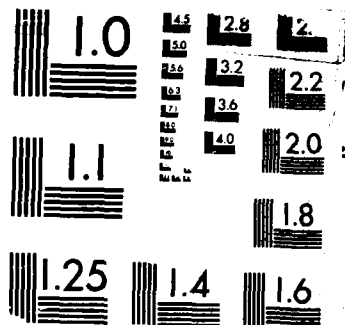
UNCLASSIFIED

HIT/LCS/TH-349 N00014-83-K-0125

F/G 12/6

NL





MICROCOPY RESOLUTION TEST CHART
 BUREAU OF STANDARDS-1963-A

LABORATORY FOR
COMPUTER SCIENCE



MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

AD-A194 454

MIT/LCS/TM-349

A PROOF OF THE KAHN PRINCIPLE FOR INPUT/OUTPUT AUTOMATA

Nancy A. Lynch
Eugene W. Stark

DTIC
ELECTE
MAR 02 1988
S G H D

January 1988

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

88 3 01 105

ADA194454

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		5. MONITORING ORGANIZATION REPORT NUMBER(S) N00014-83-K-0125 and N00014-85-K-0168	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MIT/LCS/TM-349		7a. NAME OF MONITORING ORGANIZATION Office of Naval Research/Department of Navy	
6a. NAME OF PERFORMING ORGANIZATION MIT Laboratory for Computer Science	6b. OFFICE SYMBOL (If applicable)	7b. ADDRESS (City, State, and ZIP Code) Information Systems Program Arlington, VA 22217	
6c. ADDRESS (City, State, and ZIP Code) 545 Technology Square Cambridge, MA 02139		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/DOD	8b. OFFICE SYMBOL (If applicable)	10. SOURCE OF FUNDING NUMBERS	
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22217		PROGRAM ELEMENT NO.	PROJECT NO.
		TASK NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) A PROOF OF THE KAHN PRINCIPLE FOR INPUT/OUTPUT AUTOMATA			
12. PERSONAL AUTHOR(S) Lynch, Nancy A. and Stark, Eugene, W.			
13a. TYPE OF REPORT Technical	13b. TIME COVERED FROM TO	14. DATE OF REPORT (Year, Month, Day) 1988 January	15. PAGE COUNT 9
16. SUPPLEMENTARY NOTATION The authors			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP	
		input/output automata, Kahn's fixed-point principle, determinacy	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) We use input/output automata to define a simple and general model of networks of concurrently executing, nondeterministic processes that communicate through undirectional, named ports. A notion of the input/output relation computed by a process is defined, and determinate processes are defined to be processes whose input/output relations are single-valued. We show that determinate processes compute continuous functions, and that networks of determinate processes obey Kahn's fixed-point principle. Although these results are already known, our contribution lies in the fact that the input/output automata model yields extremely simple proofs of them (the simplest we have seen), in spite of its generality. It is shown			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Judy Little, Publications Coordinator		22b. TELEPHONE (Include Area Code) (617) 253-5894	22c. OFFICE SYMBOL

A Proof of the Kahn Principle for Input/Output Automata

Nancy A. Lynch¹

M.I.T. Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139 USA

Eugene W. Stark²

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794 USA

December 17, 1987

Abstract

We use *input/output automata* to define a simple and general model of networks of concurrently executing, nondeterministic processes that communicate through unidirectional, named ports. A notion of the *input/output relation* computed by a process is defined, and *determinate* processes are defined to be processes whose input/output relations are single-valued. We show that determinate processes compute continuous functions, and that networks of determinate processes obey *Kahn's fixed-point principle*. Although these results are already known, our contribution lies in the fact that the input/output automata model yields extremely simple proofs of them (the simplest we have seen), in spite of its generality.

KEYWORDS: input/output automata, Kahn's fixed-point principle, determinacy

© Massachusetts Institute of Technology 1987

¹Research supported in part by DARPA contract N00014-83-K-0125, ONR contract N00014-85-K-0168, and NSF grant CCR-8611442.

²Research supported in part by NSF grant CCR-8702247.

A Proof of the Kahn Principle for Input/Output Automata

Nancy A. Lynch¹

M.I.T. Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139 USA

Eugene W. Stark²

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794 USA

December 17, 1987

Abstract

We use *input/output automata* to define a simple and general model of networks of concurrently executing, nondeterministic processes that communicate through unidirectional, named *ports*. A notion of the *input/output relation* computed by a process is defined, and *determinate* processes are defined to be processes whose input/output relations are single-valued. We show that determinate processes compute continuous functions, and that networks of determinate processes obey *Kahn's fixed-point principle*. Although these results are already known, our contribution lies in the fact that the input/output automata model yields extremely simple proofs of them (the simplest we have seen), in spite of its generality.

1 Introduction

In [5], Kahn describes a simple parallel programming language based on the concept of a network of concurrently executing sequential processes that can communicate by sending values over "channels." The communication primitives available to processes are sufficiently restrictive that only functional processes can be programmed. That is, each process may be viewed as computing a function from the complete history of values received on its input channels, to the complete history of values emitted on its output channels. Kahn argues that such processes in fact compute functions that are continuous with respect to a suitable complete partial order (cpo) structure on the sets of input and output histories. Moreover, a network of such processes also computes a continuous function, which can be characterized as the least fixed-point of a continuous functional associated with the network. The advantage of this least fixed-point characterization is that it permits the use of Scott's induction rule to prove properties of process networks.

Kahn's original conception of a process network has subsequently been elaborated to serve as a basis for "dataflow" models of computation. In the dataflow literature, a network of processes is typically represented by a "dataflow graph," which is a directed graph whose nodes correspond to processes, and whose arcs correspond to unidirectional FIFO communication channels between processes. The program for a process designates particular channels to be used for input or output through the use of "ports," which are names assigned by a process to each channel attached to that

¹Research supported in part by DARPA contract N00014-83-K-0125, ONR contract N00014-85-K-0168, and NSF grant CCR-8611442.

²Research supported in part by NSF grant CCR-8702247.



Availability Codes	
Dist	Avail and/or Special
A-1	

process. In contrast to Kahn's original model, both functional and nonfunctional processes are of interest in dataflow computation. Although it is straightforward to give an operational semantics for such networks by describing the flow of data values through them, it is unfortunately the case that Kahn's denotational semantics for networks of functional processes is not known to have an equally elegant generalization to networks of processes with non-functional behaviors. Brock and Ackerman [1] have shown that naive generalizations, in which relations, rather than functions, are used to represent the input/output behavior of processes, fail to be consistent with the intuitive operational model of network execution. An extensive literature has arisen from attempts to resolve the so-called "Brock-Ackerman anomaly." Although we cannot adequately review this literature here, the reader may refer to the recent papers [4,6,9] for references to earlier work.

Kahn did not give a proof of the consistency of his fixed-point principle with respect to an operational semantics. However, Kahn's principle is similar to results that had already been proved [2] for recursive program schemes, and thus was generally accepted without an explicit proof. In the search for extensions to the non-functional case, though, consistency proofs are essential, since it is fairly easy to define denotational "semantics" which, although seemingly plausible, do not agree with an intuitively correct operational semantics. Recently, some attention has been paid to the problem of establishing the Kahn principle as a theorem about an operational model. Faustini [3] defines a reasonably general model of networks of nondeterministic processes. Using some game-theoretic ideas, Faustini defines a subclass of networks of functional processes, and shows that such networks obey the Kahn principle. Stark [9] defines a class of nondeterministic processes, through axioms that constrain the structure of processes viewed as a kind of generalized transition system. "Kahn processes" are defined to be processes whose underlying transition systems obey an additional Church-Rosser-like property. Stark shows that the Kahn principle can be derived from the axioms. Gaifman and Pratt [4], and Rabinovich [8] show that the Kahn principle holds for the "pomset" model.

Although the technical complexities of the three papers [4,8,9] make anything other than qualitative comparisons difficult, all seem to be talking about essentially similar sets of ideas. Each of the proofs involves the use of the properties:

1. A process is capable of accepting any input at any time.
2. Production of output by a process depends only on previously received input, and not on input received later than or simultaneously with the output.
3. If the input history of a process in one computation is consistent with its input history in another computation, then the output histories in the two computations are also consistent.

These three properties are used in an inductive argument to show that a network must produce output less than or equal to the output specified by the Kahn principle. The additional property:

4. A process can always make progress toward a complete computation, regardless of the input received.

is used to establish that a network must produce at least as much output as that specified by the Kahn principle.

In this paper, we prove the consistency of the Kahn principle with respect to an operational model based on the "input/output automata" of Lynch and Tuttle [7]. Our proof shares with

others the four central ideas listed above, but has the advantage of being extremely simple (the simplest we have yet seen). In part, this simplicity is attained because we are able to make use of two powerful general theorems (Lemma 1 and Proposition 2) about input/output automata. Our model is more general than Faustini's [3], since we do not make any concrete assumption about the structure of "channel buffers." Faustini postulates channel buffers whose states are sequences of messages in transit. In contrast, we think of each process as containing, as components of its state, the buffers for the channels from which it takes its input. We also do not require for our definitions and proofs the game theory used by Faustini. Our work can be seen as complementary in a sense to that of Stark [9]. Whereas the latter work can be viewed as a search for as weak a condition as possible on nondeterministic processes, from which the Kahn principle can be proved, our results show that the simple restriction to "determinate" processes (those with single-valued input/output relations) is already an extremely strong constraint, from which the Kahn principle follows almost automatically.

Even though the truth of the Kahn principle is not really in doubt, we believe it is important to search for semantic models in which the principle can be proved as simply and generally as possible. Since this principle is perhaps the simplest and most elegant result we have to date in the theory of concurrency, it seems reasonable to expect that any purportedly useful semantic model should admit a simple proof of it. The ultimate goals of the search would be the identification of a minimal set of properties that a model of nondeterministic process networks must have if the Kahn principle is to hold, and a determination of the extent to which the theory of functional processes can be usefully generalized.

2 Input/Output Automata

An *action signature* is a triple $A = (A^{\text{in}}, A^{\text{out}}, A^{\text{int}})$, where the sets A^{in} , A^{out} , and A^{int} are pairwise disjoint. The elements of A^{in} are called *input actions*, those of A^{out} are called *output actions*, and those of A^{int} , *internal actions*. We use the same symbol A to denote both an action signature and the set $A^{\text{in}} \cup A^{\text{out}} \cup A^{\text{int}}$ of all its actions.

An *input/output automaton* is a tuple $M = (A, Q, Q^\circ, T, \sim)$, where

- A is an action signature.
- Q is a set of *states*.
- $Q^\circ \subseteq Q$ is a distinguished set of *start states*.
- $T \subseteq Q \times A \times Q$ is a set of *transitions*, with the property that for all $q \in Q$ and all input actions a , there exists a transition (q, a, r) in T .
- \sim is an equivalence relation on the set $(A^{\text{out}} \cup A^{\text{int}})$ of non-input actions, such that the number of equivalence classes of \sim is at most countable.

If $(q, a, r) \in T$, and T is clear from the context, then we write $q \xrightarrow{a} r$. An action a is said to be *enabled* in state q if there exists a state r such that $q \xrightarrow{a} r$. The definition of an input/output automaton requires that all input actions be enabled in every state.

A comment is in order concerning the equivalence relation \sim . We use input/output automata not just to model single processes, but also systems of concurrently executing processes. When we

model a system of processes, we are interested only in "fair" computations, that is, in computations in which no process that desires to execute is forever prevented from doing so. To impose the requirement of fairness, we need a certain amount of information about the correspondence between actions and processes. The equivalence relation \sim provides this information, in the sense that we think of each equivalence class of \sim as the set of actions of a single process that should receive fair treatment.

An *execution fragment* of an input/output automaton is either a finite sequence of the form

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-1}} q_n,$$

or an infinite sequence of the form

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots,$$

where for each $k \geq 0$, we require that $q_k \xrightarrow{a_{k+1}} q_{k+1} \in T$. An *execution* is an execution fragment whose first state q_0 is a start state.

A finite execution fragment

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-1}} q_n,$$

is *fair* if no non-input actions are enabled in state q_n . An infinite execution fragment

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots$$

is *fair* if, for every \sim -equivalence class C of actions, either there exist infinitely many $k \geq 0$ with $a_k \in C$, or else there exist infinitely many $k \geq 0$ for which no action in C is enabled in state q_k .

If U is any set, then let U^∞ denote the set of all finite and infinite sequences of elements of U . If A is an action signature, then we call A^∞ the set of *action sequences* for A . If σ is an action sequence, and U is a set, then the *restriction* of σ to U is the subsequence $\sigma|U$ of σ consisting only of those actions that are in U . If M is an input/output automaton, then the *schedule* of an execution fragment of M is the sequence of actions appearing in that fragment. The set $\text{finscheds}(M)$ of *finite schedules* of M is the set of all schedules of finite executions of M . The set $\text{fairscheds}(M)$ of *fair schedules* of M is the set of all schedules of fair executions of M .

Lemma 1 *Let M be an input/output automaton, and suppose $\sigma \in \text{finscheds}(M)$. Then given any action sequence ρ consisting only of input actions, there exists a sequence τ such that $\sigma\tau \in \text{fairscheds}(M)$, and such that $\tau|A^{\text{in}} = \rho$.*

Proof – We first claim that given any state $q \in Q$, and sequence ρ consisting only of input actions, there exists a fair execution fragment, starting from state q and having schedule τ , such that $\tau|A^{\text{in}} = \rho$. This fair execution fragment can be obtained by a dovetailing construction in which actions in ρ are interleaved with actions from the various equivalence classes of \sim . The condition that every input action is enabled in every state of an input/output automaton ensures that actions in ρ can be executed whenever required. The condition that the set of equivalence classes of \sim is at most countable ensures that the dovetailing can be carried out in such a way that the resulting execution fragment is fair.

It is now easy to prove our result. Given $\sigma \in \text{finscheds}(M)$, obtain a finite execution

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-1}} q_n$$

with schedule σ . Given a sequence ρ consisting only of input actions, apply the claim of the previous paragraph to obtain a fair execution fragment, starting from state q_n and having schedule τ , such that $\tau|_{A_i^{\text{in}}} = \rho$. Concatenating the finite execution with schedule σ with the fair execution fragment with schedule τ yields a fair execution with schedule $\sigma\tau$, thus showing $\sigma\tau \in \text{fairscheds}(M)$. ■

Suppose I is a finite or countably infinite index set. A collection $\mathcal{A} = \{A_i : i \in I\}$ of action signatures is called *compatible* if for all $i, j \in I$ with $i \neq j$ we have $A_i^{\text{out}} \cap A_j^{\text{out}} = \emptyset$ and $A_i^{\text{int}} \cap A_j = \emptyset$. If \mathcal{A} is compatible, then the *composition* of \mathcal{A} is the action signature $\prod \mathcal{A} = (A^{\text{in}}, A^{\text{out}}, A^{\text{int}})$, where $A^{\text{out}} = \bigcup_{i \in I} A_i^{\text{out}}$, $A^{\text{in}} = (\bigcup_{i \in I} A_i^{\text{in}}) \setminus A^{\text{out}}$, and $A^{\text{int}} = \bigcup_{i \in I} A_i^{\text{int}}$.

A collection $\mathcal{M} = \{M_i : i \in I\}$ of input/output automata, where M_i has signature A_i , is called *compatible* if the collection $\mathcal{A} = \{A_i : i \in I\}$ of action signatures is compatible. If \mathcal{M} is compatible, then the *composition* of \mathcal{M} is the quintuple $\prod \mathcal{M} = (A, Q, Q^o, T, \sim)$, where

- $A = \prod \mathcal{A}$.
- $Q = \prod_{i \in I} Q_i$.
- $Q^o = \prod_{i \in I} Q_i^o$.
- T is the set of all $((q_i : i \in I), a, (r_i : i \in I))$ such that for all $i \in I$, if $a \in A_i$, then $(q_i, a, r_i) \in T_i$, and if $a \notin A_i$, then $r_i = q_i$.
- $\sim = \bigcup_{i \in I} \sim_i$.

It is easy to see that the compatibility condition ensures that $\prod \mathcal{M}$ is an input/output automaton.

The following result characterizes the set of finite or fair schedules of $\prod \mathcal{M}$ in terms of the sets of finite or fair schedules of the M_i . A proof can be found in [7].

Proposition 2 Suppose $\mathcal{M} = \{M_i : i \in I\}$ is a compatible collection of input/output automata. For each $i \in I$, let A_i be the action signature of M_i . Then

1. Suppose σ is a finite sequence of actions from $\prod \{A_i : i \in I\}$. Then $\sigma \in \text{finscheds}(\prod \mathcal{M})$ iff $\sigma|_{A_i} \in \text{finscheds}(M_i)$ for all $i \in I$.
2. $\sigma \in \text{fairscheds}(\prod \mathcal{M})$ iff $\sigma|_{A_i} \in \text{fairscheds}(M_i)$ for all $i \in I$.

3 Port Automata

Let V be a set of *data values*. A *port signature* is an action signature A , whose sets of input and output actions have the particular form $A^{\text{in}} = P^{\text{in}} \times V$ and $A^{\text{out}} = P^{\text{out}} \times V$, with P^{in} and P^{out} disjoint and at most countable. The elements of P^{in} and P^{out} are called *input ports* and *output ports*, respectively. If $a = (p, v) \in A^{\text{in}} \cup A^{\text{out}}$, then we write $\text{port}(a)$ for the port component p , and $\text{value}(a)$ for the value component v , of a . A *port automaton* is an input/output automaton whose action signature is a port signature.

Suppose $\mathcal{A} = \{A_i : i \in I\}$ is a compatible collection of port signatures. Then the composition $\prod \mathcal{A}$ is also a port signature, with output port set $P^{\text{out}} = \bigcup_{i \in I} P_i^{\text{out}}$ and input port set $P^{\text{in}} = (\bigcup_{i \in I} P_i^{\text{in}}) \setminus P^{\text{out}}$. It follows that the composition of a compatible collection of port automata is also a port automaton.

The composition of a compatible collection of port automata models a network of communicating, concurrently executing, component processes. Communication between components in such a network occurs when an output transition of one component, with a particular port and data value, occurs simultaneously with input transitions, with the same port and data value, for a number of other components. We allow arbitrary "fanout" in the sense that a single action may be shared by more than two components, as long as it is an output action for at most one of them. This is a bit more general than the usual definition of "linking" in the dataflow literature, in which each port of a process may be connected with at most one port of another process. We do not have any formal notion of "input buffers" or "channel processes." Rather, we think of a buffer for each input port of a process as already incorporated into the state of that process.

If P is a set of ports, then a *history over P* is a function $H : P \rightarrow V^\infty$. Let $\text{Hist}(P)$ denote the set of all histories over P . If A is a port signature, then each sequence σ in A^∞ determines a corresponding history $H_\sigma \in \text{Hist}(P^{\text{in}} \cup P^{\text{out}})$, defined by

$$H_\sigma(p) = \text{value}(\sigma | \{a \in A^{\text{in}} \cup A^{\text{out}} : \text{port}(a) = p\}),$$

where we have extended the 'value' notation to sequences $\sigma = a_1 a_2 \dots \in (A^{\text{in}} \cup A^{\text{out}})^\infty$, by defining $\text{value}(\sigma) = \text{value}(a_1)\text{value}(a_2)\dots$. The restrictions $H_\sigma^{\text{in}} = H_\sigma | P^{\text{in}}$ and $H_\sigma^{\text{out}} = H_\sigma | P^{\text{out}}$ to the sets of input and output ports, respectively, are called the *input history* and *output history* of σ . The *input/output relation* of a port automaton M is the set $\text{Reln}(M)$ of all pairs $(H_\sigma^{\text{in}}, H_\sigma^{\text{out}})$ with $\sigma \in \text{fairscheds}(M)$.

It is important for our purposes that the sets A^∞ and V^∞ , and the set $\text{Hist}(P)$ of all histories $H : P \rightarrow V^\infty$, form *algebraic, directed-complete posets*¹ when equipped with suitable partial orderings. The ordering of interest on A^∞ and V^∞ is the prefix ordering, and on $\text{Hist}(P)$ it is the ordering \sqsubseteq obtained componentwise from the prefix ordering on V^∞ . The finite elements of A^∞ and V^∞ are the finite sequences, and the finite elements of $\text{Hist}(P)$ are exactly those functions from P to V^* that map all but a finite subset of P to the empty sequence. Moreover, the map that takes a sequence $\sigma \in A^\infty$ to the corresponding history H_σ is continuous, and maps finite sequences to finite histories. Finally, note that the assumption that P is at most countable ensures that every history $H \in \text{Hist}(P)$ is H_σ for some sequence $\sigma \in A^\infty$.

4 Determinacy

A port automaton M is *determinate* if its input/output relation $\text{Reln}(M)$ is single-valued, hence is the graph of a function

$$\text{Fun}(M) : \text{Hist}(P^{\text{in}}) \rightarrow \text{Hist}(P^{\text{out}}).$$

Lemma 3 *Suppose M is determinate. Suppose $\sigma \in \text{finscheds}(M)$ and $\tau \in \text{fairscheds}(M)$ are such that $H_\sigma^{\text{in}} \sqsubseteq H_\tau^{\text{in}}$. Then $H_\sigma^{\text{out}} \sqsubseteq H_\tau^{\text{out}}$.*

¹A subset U of a partially ordered set (poset) (D, \sqsubseteq) is *directed* if it is nonempty and every pair of elements of U has an upper bound in U . The poset (D, \sqsubseteq) is *directed-complete* if it has a least element, and every directed subset U of D has a supremum $\bigsqcup U \in D$. A function between directed-complete posets is called *continuous* if it preserves suprema. If (D, \sqsubseteq) is directed-complete, then an element $e \in D$ is called *finite* (also *isolated*, or *compact*) if whenever $U \subseteq D$ is directed, and $e \sqsubseteq \bigsqcup U$, then $e \sqsubseteq d$ for some $d \in U$. It is *algebraic* if every element $d \in D$ is the supremum of the set of all finite $e \in D$ with $e \sqsubseteq d$.

Proof – By Lemma 1, σ extends to a schedule ρ in $\text{fairscheds}(M)$, such that $H_\rho^{\text{in}} = H_\tau^{\text{in}}$. By determinacy, we must have $H_\rho^{\text{out}} = H_\tau^{\text{out}}$. Since $H_\sigma^{\text{out}} \subseteq H_\rho^{\text{out}}$ by construction, it follows that $H_\sigma^{\text{out}} \subseteq H_\tau^{\text{out}}$. ■

Lemma 4 Suppose M is determinate, with $\text{Fun}(M) = f$. Then $H_\sigma^{\text{out}} \subseteq f(H_\sigma^{\text{in}})$ for all $\sigma \in \text{finscheds}(M)$.

Proof – Given $\sigma \in \text{finscheds}(M)$, we may use Lemma 1 to extend σ to $\tau \in \text{fairscheds}(M)$, with $H_\tau^{\text{in}} = H_\sigma^{\text{in}}$. Then $H_\sigma^{\text{out}} \subseteq H_\tau^{\text{out}}$ by Lemma 3, and $H_\tau^{\text{out}} = f(H_\tau^{\text{in}})$ by the fact that $\tau \in \text{fairscheds}(M)$. Since $H_\tau^{\text{in}} = H_\sigma^{\text{in}}$, $f(H_\tau^{\text{in}}) = f(H_\sigma^{\text{in}})$. Thus, $H_\sigma^{\text{out}} \subseteq f(H_\sigma^{\text{in}})$. ■

Theorem 1 If M is determinate, then $\text{Fun}(M)$ is continuous.

Proof – We first show monotonicity. Suppose $\sigma, \tau \in \text{fairscheds}(M)$, with $H_\sigma^{\text{in}} \subseteq H_\tau^{\text{in}}$. Then $H_\rho^{\text{in}} \subseteq H_\tau^{\text{in}}$ holds for all finite prefixes ρ of σ , so by Lemma 3, $H_\rho^{\text{out}} \subseteq H_\tau^{\text{out}}$ holds for all finite prefixes ρ of σ . It follows that $H_\sigma^{\text{out}} \subseteq H_\tau^{\text{out}}$.

Next, we show continuity. Suppose $\Sigma \subseteq \text{fairscheds}(M)$, such that the collection $\{H_\sigma^{\text{in}} : \sigma \in \Sigma\}$ is directed, with supremum H^{in} . By Lemma 1 and the fact that H^{in} is the history of some sequence consisting only of input actions, we know there exists a schedule $\tau \in \text{fairscheds}(M)$ with $H_\tau^{\text{in}} = H^{\text{in}}$. Then by monotonicity, $H_\sigma^{\text{out}} \subseteq H_\tau^{\text{out}}$ for all $\sigma \in \Sigma$. This implies that the collection $\{H_\sigma^{\text{out}} : \sigma \in \Sigma\}$ is directed, hence has a supremum $H^{\text{out}} \subseteq H_\tau^{\text{out}}$. We claim that $H_\tau^{\text{out}} \subseteq H^{\text{out}}$. By the continuity of the map that takes each action sequence to the corresponding history, it suffices to show that $H_\rho^{\text{out}} \subseteq H^{\text{out}}$ for all finite prefixes ρ of τ . But if ρ is a finite prefix of τ , then $H_\rho^{\text{in}} \subseteq H^{\text{in}}$, hence $H_\rho^{\text{in}} \subseteq H_\sigma^{\text{in}}$ for some $\sigma \in \Sigma$ by the finiteness of H_ρ^{in} . Thus $H_\rho^{\text{out}} \subseteq H_\sigma^{\text{out}}$ by Lemma 3, and therefore $H_\rho^{\text{out}} \subseteq H^{\text{out}}$. ■

5 The Kahn Principle

Let $\mathcal{A} = \{A_i : i \in I\}$ be a compatible collection of port signatures. Let P denote the set of ports of $\prod \mathcal{A}$, and for each $i \in I$, let P_i denote the set of ports of A_i . Suppose $\mathcal{F} = \{f_i : i \in I\}$ is a collection of continuous functions, where for each $i \in I$,

$$f_i : \text{Hist}(P_i^{\text{in}}) \rightarrow \text{Hist}(P_i^{\text{out}}).$$

The *network equations* associated with \mathcal{F} are the equations (in the unknown history $H \in \text{Hist}(P)$):

$$H|P_i^{\text{out}} = f_i(H|P_i^{\text{in}}) \quad (i \in I).$$

The *network functional* associated with \mathcal{F} is the function

$$\Phi : [\text{Hist}(P^{\text{in}}) \rightarrow \text{Hist}(P)] \rightarrow [\text{Hist}(P^{\text{in}}) \rightarrow \text{Hist}(P)]$$

that takes each continuous function

$$f : \text{Hist}(P^{\text{in}}) \rightarrow \text{Hist}(P)$$

to the function

$$\Phi(f) : \text{Hist}(P^{\text{in}}) \rightarrow \text{Hist}(P)$$

defined by

$$\Phi(f)(H^{\text{in}})|P^{\text{in}} = H^{\text{in}}, \quad \Phi(f)(H^{\text{in}})|P_i^{\text{out}} = f_i(f(H^{\text{in}})|P_i^{\text{in}}).$$

The compatibility condition on \mathcal{A} ensures that Φ is well-defined, and it is straightforward to verify that $\Phi(f)$ is continuous whenever f is continuous.

The following result can be proved by standard techniques in the theory of cpo's (see, e.g. [5], Section 3).

Proposition 5 *Suppose port signatures \mathcal{A} and functions \mathcal{F} are as above. Then the network functional Φ associated with \mathcal{F} is continuous, hence has a least fixed point $\mu\Phi$. Moreover, $\mu\Phi$ takes each history $H^{\text{in}} \in \text{Hist}(P^{\text{in}})$ to the least history $H \in \text{Hist}(P)$ such that $H|P^{\text{in}} = H^{\text{in}}$, and such that H satisfies the network equations associated with \mathcal{F} .*

Theorem 2 (Kahn Principle) *Suppose $\mathcal{M} = \{M_i : i \in I\}$ is a compatible collection of determinate port automata, let $\mathcal{F} = \{\text{Fun}(M_i) : i \in I\}$, and let Φ be the network functional associated with \mathcal{F} . Then $\prod \mathcal{M}$ is determinate, and $\text{Fun}(\prod \mathcal{M})$ satisfies*

$$\text{Fun}(\prod \mathcal{M})(H^{\text{in}}) = \mu\Phi(H^{\text{in}})|P^{\text{out}}$$

for all $H^{\text{in}} \in \text{Hist}(P^{\text{in}})$.

Proof – Let $f_i = \text{Fun}(M_i)$ for each $i \in I$. By Proposition 5, it suffices to show that for each schedule $\sigma \in \text{fairscheds}(\prod \mathcal{M})$, the history H_σ is the least history $H \in \text{Hist}(P)$ such that $H|P^{\text{in}} = H_\sigma^{\text{in}}$, and such that H satisfies the network equations associated with \mathcal{F} .

Suppose $\sigma \in \text{fairscheds}(\prod \mathcal{M})$. Since $\sigma|A_i \in \text{fairscheds}(M_i)$ by Proposition 2, it follows that for each $i \in I$, $H_\sigma|P_i^{\text{out}} = H_\sigma|A_i^{\text{out}} = f_i(H_\sigma|A_i^{\text{in}}) = f_i(H_\sigma|P_i^{\text{in}})$. Thus, the network equations are satisfied by H_σ .

It remains to be shown that if H is any history with $H^{\text{in}} = H_\sigma^{\text{in}}$ such that H satisfies the network equations, then $H_\sigma \sqsubseteq H$. It suffices to show that $H_\rho \sqsubseteq H$ for all finite prefixes ρ of σ . We proceed by induction on the length $|\rho|$ of ρ . The basis, $|\rho| = 0$, is immediate. For the induction step, let $\rho = \rho'a$ where $a \in A$ and $H_{\rho'} \sqsubseteq H$. There are three cases:

(Case $a \in A^{\text{int}}$) Then $H_\rho = H_{\rho'} \sqsubseteq H$.

(Case $a \in A^{\text{in}}$) Since $H_\sigma^{\text{in}} = H^{\text{in}}$, we have $H_\rho^{\text{in}} \sqsubseteq H^{\text{in}}$. Then $H_{\rho'} \sqsubseteq H$ and $H_\rho^{\text{in}} \sqsubseteq H^{\text{in}}$ together imply $H_\rho \sqsubseteq H$.

(Case $a \in A^{\text{out}}$) Then $a \in A_i^{\text{out}}$ for some $i \in I$, so that $H_\rho|P_i^{\text{in}} = H_{\rho'}|P_i^{\text{in}}$. By Proposition 2 and Lemma 4 we know that $H_\rho|P_i^{\text{out}} \sqsubseteq f_i(H_\rho|P_i^{\text{in}}) = f_i(H_{\rho'}|P_i^{\text{in}})$. But $H_{\rho'}|P_i^{\text{in}} \sqsubseteq H|P_i^{\text{in}}$, hence $f_i(H_{\rho'}|P_i^{\text{in}}) \sqsubseteq f_i(H|P_i^{\text{in}}) = H|P_i^{\text{out}}$ by the monotonicity of f_i and the assumption that H satisfies the network equations. Thus, $H_\rho|P_i^{\text{out}} \sqsubseteq H|P_i^{\text{out}}$. This fact, together with $H_{\rho'} \sqsubseteq H$, implies $H_\rho \sqsubseteq H$. ■

6 Conclusion

We have used input/output automata to define a rather general model of networks of nondeterministic processes. A notion of the input/output relation computed by a process has been defined, and used to define the class of determinate (or functional) processes. We have shown that determinacy is a very strong property, from which it follows almost immediately that the functions computed by determinate processes are continuous, and that networks of determinate processes obey the Kahn principle.

References

- [1] J. D. Brock and W. B. Ackerman. Scenarios: a model of non-determinate computation. In *Formalization of Programming Concepts*, pages 252–259, Springer-Verlag. Volume 107 of *Lecture Notes in Computer Science*, 1981.
- [2] J. M. Cadiou. *Recursive Definitions of Partial Functions and Their Computations*. PhD thesis, Stanford University, 1972.
- [3] A. A. Faustini. An operational semantics for pure dataflow. In *Automata, Languages, and Programming, 9th Colloquium*, pages 212–224, Springer-Verlag. Volume 140 of *Lecture Notes in Computer Science*, 1982.
- [4] H. Gaifman and V. Pratt. Partial order models of concurrency and the computation of functions. In *Symposium on Logic in Computer Science*, pages 72–85, June 1987.
- [5] G. Kahn. The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information Processing 74*, North-Holland, 1974.
- [6] J. N. Kok. A fully abstract semantics for data flow nets. pages 351–368, Springer-Verlag. Volume 259 of *Lecture Notes in Computer Science*, 1987.
- [7] N. A. Lynch and M. Tuttle. *Hierarchical Correctness Proofs for Distributed Algorithms*. Technical Report MIT/LCS/TR-387, M. I. T. Laboratory for Computer Science, April 1987.
- [8] A. Rabinovich. Pomset semantics is consistent with data flow semantics. *EATCH Bulletin*, 107–117, June 1987.
- [9] E. W. Stark. Concurrent transition system semantics of process networks. In *Fourteenth ACM Symposium on Principles of Programming Languages*, pages 199–210, January 1987.

OFFICIAL DISTRIBUTION LIST

Director 2 copies
Information Processing Techniques Office
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Office of Naval Research 2 copies
800 North Quincy Street
Arlington, VA 22217
Attn: Dr. R. Grafton, Code 433

Director, Code 2627 6 copies
Naval Research Laboratory
Washington, DC 20375

Defense Technical Information Center 12 copies
Cameron Station
Alexandria, VA 22314

National Science Foundation 2 copies
Office of Computing Activities
1800 G. Street, N.W.
Washington, DC 20550
Attn: Program Director

Dr. E.B. Royce, Code 38 1 copy
Head, Research Department
Naval Weapons Center
China Lake, CA 93555

END

DATE

FILMED

8-88

DTIC